A Quadratic Time-Space Tradeoff for Unrestricted Deterministic Decision Branching Programs

Nandakishore Santhi
nsanthi@ucsd.edu

Alexander Vardy
vardy@kilimanjaro.ucsd.edu

Department of Electrical and Computer Engineering,
University of California San Diego,
9500 Gilman Drive, MC 0407, La Jolla, CA-92093-0407, USA.

Abstract

The branching program is a fundamental model of nonuniform computation, which conveniently captures both time and space restrictions. In this paper we prove a quadratic expected time-space tradeoff of the form $\overline{T}\overline{S} = \Omega\left(\frac{n^2}{q}\right)$ for q-way deterministic decision branching programs, where $q \geqslant 2$. Here $\overline{T}$ is the expected computation time and $\overline{S}$ is the expected space, when all inputs are equally likely. This bound is to our knowledge, the first such to show an exponential size requirement whenever $\overline{T} = \mathcal{O}(n^2)$. Previous exponential size tradeoffs for Boolean decision branching programs were valid for time-restricted models with $T = o(n \log_2 n)$. Proving quadratic time-space tradeoffs for unrestricted time decision branching programs has been a major goal of recent research – this goal has already been achieved for multiple-output branching programs a few decades ago. The decision branching programs we consider are related to families of good linear codes.

Our results also imply the first quadratic time-space tradeoffs for Boolean decision branching programs verifying circular convolution, matrix-vector multiplication and discrete Fourier transform. A quadratic tradeoff is the largest possible for all these problems. Using the constructive family of Justesen codes which are asymptotically good, we also demonstrate a constructive Boolean decision function which has a quadratic expected time-space tradeoff in the Boolean deterministic decision branching program model.

For q-way programs where q is a constant, the tradeoff results derived here for decision functions verifying various functions are order-comparable to previously known tradeoff bounds for calculating the corresponding multiple-output functions. In deriving these bounds we use several bounding techniques and introduce a few new ideas. These include a particular measure of progress which is specific to the decision function considered, partitioning the computational paths into disjoint sets and obtaining tradeoffs for each class separately and extensive use of linear constraints to obtain probability bounds.

I. Introduction

The branching program is a fundamental model of nonuniform computation, which conveniently captures both time and space restrictions. See Section II-B for a formal definition. Proving lower bounds on computational resources (such as time and space) required to compute a specific (explicit) function in a general computational model is a notoriously difficult task. Topics of interest in this direction are both the bounds themselves as well as the methods for obtaining these bounds.

There are many different types of branching programs, and it is important to distinguish between them. The first major distinction, between *decision (single-output)* branching programs and *multi-output* branching programs, has to do with how the output of the program is produced. In the case of decision branching programs the output is a single bit: each sink node is labeled by either 0 or 1, and the output of the program is simply the value labeling the sink node reached. In the case of multi-output branching programs, the output is a sequence of $m > 1$ values, and each node in $\mathcal{B}$ is allowed to assign (write) at most

This work was supported in part by the National Science Foundation and by the California Institute of Telecommunications and Information Technology at UCSD.

one of these m values. In general, it appears to be *much harder* to prove lower bounds on time and space for decision branching programs than for multi-output branching programs. The second distinction has to do with *large domains* versus *small (boolean) domains*. In the large-domain case, each input variable takes values from a set whose size grows with the length of the input (the number of variables). In the small-domain case, the input variables take values in a fixed set of constant size — the set $\{0,1\}$ for boolean domains. Again, it appears to be more difficult to prove time-space lower bounds for boolean (2-way) branching programs than for branching programs over large domains. Finally, one distinguishes between general, unrestricted, branching programs and branching programs that are restricted in some important way. Common restrictions include *read-k* (each input variable is accessed at most k times) and *oblivious* (input variables are read in the same order along all paths) branching programs. Obviously, it is much harder to establish lower bounds for the more powerful, unrestricted, model than for branching programs that are read-k, oblivious, or otherwise restricted.

In this paper we seek to answer one of the fundamental questions in theoretical computer science:

"Can the verification of a particular computation be as complex as performing the computation itself?"

We will also be concerned with the most difficult case — *unrestricted boolean decision* branching programs — where the state of knowledge concerning lower bounds on time and space for concrete functions was rather pathetic until recently. In fact, no such bounds *at all* were known until the groundbreaking work of Beame, Jayram, and Saks [BST98, BJS01]. In [BST98] and [BJS01], the authors extended the techniques of Borodin, Razborov, and Smolensky [BRS83] to prove the first (barely) nontrivial bound of this kind. They exhibited a problem in P, based upon quadratic forms over a finite field (cf. [BRS83]), for which any sub-exponential size branching program requires time at least $(1+\varepsilon)n$, where n is the input length and $\varepsilon > 0$ is a constant. In a remarkable breakthrough, Ajtai [Ajt99, Ajt98] constructed a polynomial-time computable Boolean function (also based on quadratic forms) for which any sub-exponential size branching program requires *super-linear time*. In another breakthrough paper, Beame, Saks, Sun, and Vee [BSS03] improved upon Ajtai's results by establishing (for the quadratic-form and element-distinctness boolean functions), a time-space tradeoff of the form $T = \Omega\big(n\sqrt{\log(n/S)/\log\log(n/S)}\,\big)$, which furthermore extends to randomized branching programs with (two-sided) error. In the last couple of years, there was more work along these lines (see [SW03, Juk02] and other recent papers). Nevertheless, the number of concrete decision problems for which super-linear time-space tradeoffs are known in the unrestricted boolean branching program model remains preciously small. Moreover it should be noted that none of these tradeoff results are valid when $T = \Omega(n \log n)$ – therefore all the above mentioned results are for *time-restricted* branching programs. In fact, proving quadratic time-space tradeoffs for unrestricted time decision branching programs has been a major goal of recent research, as mentioned for example by Beame, Saks, Sun and Vee in [BSS03] – a similar goal having been long since achieved for multiple-output branching programs [BC82, Yes84, Abr91].

Herein we prove a quadratic expected time-space tradeoff of the form $\overline{TS} = \Omega\big(\frac{n^2}{q}\big)$ and valid for unrestricted (including time) q-way deterministic decision branching programs, where $q \geqslant 2$. Here $\overline{T}$ is the expected computation time and $\overline{S}$ is the expected space. This bound is to our knowledge, the first such to show an exponential size requirement which holds for $\overline{T} = \mathcal{O}(n^2)$. As mentioned before, previous exponential size tradeoffs for Boolean decision branching programs were valid when $T = o(n \log_2 n)$. The branching programs we consider are related to families of good linear codes. The tradeoff results shown here for decision functions are order-comparable (for q-way programs where q is a constant) to the tradeoff bounds obtained by Santhi and Vardy [SV06] for a closely related multiple-output function.

We also remark that using the constructive family of Justesen codes which are asymptotically good, one may also demonstrate a constructive Boolean decision function which has a quadratic expected time-space tradeoff in the unrestricted deterministic Boolean decision branching program model.

Our results also imply the first ever quadratic time-space tradeoffs for unrestricted deterministic Boolean

Table I *Various Bounds for deterministic branching programs are compared.* C *is an* $(n,k,d)_q$ *code,* E_{C} *is its encoding function,* $f_{\mathrm{C}}^{\perp}$ *is the dual syndrome-vector function and* $f_{\mathrm{C},\gamma}$ *is the partial verifier for the dual syndrome-vector (code-coset membership). These are the tightest among all such known distance bounds.*

Type of branching program	Tradeoff	Computed function
q-way, multi-output, unrestricted deterministic	$\overline{TS} = \Omega(n^2 \log_2 q)$	n-length CONV, n-length MVMUL and n-point DFT (see [Abr91])
q-way, multi-output, only restriction: $T = o(n \log_2 n)$, worst case complexity, any C	$d = \mathcal{O}\left(\frac{T^2}{k}\left(\frac{S}{k \log_2 q}\right)^{\frac{k}{2T}}\right)$	E_{C} (see [BM05, San06])
q-way, multi-output, unrestricted deterministic, linear C	$d \leqslant \frac{12\overline{T}\left(\overline{S}+\log_2 \overline{T}+6\right)}{k \log_2 q} + 1$	$f_{\mathrm{C}}^{\perp}$ (see [SV06])
q-way, decision, unrestricted deterministic, linear C	$d = \mathcal{O}\left(\frac{q\overline{TS}}{k}\right)$	$f_{\mathrm{C},\gamma}$ (here)

Table II *New bounds for decision branching programs (partially) verifying some fundamental functions as derived in this article. Previous corresponding non-linear (worst-case) time-space tradeoff results known for* $\zeta_{\mathrm{CONV},\gamma}$ *and* $\zeta_{\mathrm{MVMUL},\gamma}$ *were for time-restricted branching programs with worst case time,* $T = o(n \log_2 n)$*. No non-trivial corresponding result is known for* $\zeta_{\mathrm{DFT},\gamma}$*. The tabulated results are derived using the distance bounds from the final row of Table I.*

Type of branching program	Tradeoff	Computed function
q-way, decision, unrestricted deterministic	$\overline{TS} = \Omega(\frac{n^2}{q})$	$\Theta(n)$-length q-ary $\zeta_{\mathrm{MVMUL},\gamma}$
q-way, decision, unrestricted deterministic	$\overline{TS} = \Omega(\frac{n^2}{q})$	$\Theta(n)$-length q-ary $\zeta_{\mathrm{CONV},\gamma}$
Boolean, decision, unrestricted deterministic	$\overline{TS} = \Omega(n^2 \log n)$	$\Theta(n)$-point $\zeta_{\mathrm{DFT},\gamma}$

decision branching programs which partial-verify circular convolution, matrix-vector multiplication and discrete Fourier transform. A quadratic tradeoff is the largest possible for all these problems, because trivial programs can be constructed otherwise. The tradeoff results obtained here for decision functions verifying fundamental operations are order-comparable (for q-way programs where q is a constant) to the tradeoff bounds derived by Abrahamson [Abr91] for calculating the corresponding multiple-output functions.

In deriving these bounds we use several bounding techniques and introduce a few new ones. These include a particular measure of progress which is specific to the decision function considered, partitioning the computational paths into disjoint sets and obtaining tradeoffs for each class separately, the concept of partial-verification and extensive use of linear constraints to obtain probability bounds.

II. Mathematical Background and Definitions

A. Branching Programs

The *branching program* (BP) has emerged as the standard general model for nonuniform sequential computation. This model is of fundamental importance (perhaps, second only to the Turing machine); it was introduced some 50 years ago by Lee [Lee59] and has been extended, refined and extensively studied in a number of papers since then [Kuz76, BC82, Weg87, Abr91, Ajt98, Ajt99, BJS01, BSS03, BST98, BC82, BRS83, Weg00].

The branching program model imposes no structure on the computation and allows any pattern of access to the input. Nevertheless, this model is strong enough to efficiently simulate many other models

of sequential computation, such as RAMs with *arbitrary instruction sets* [BC82]. RAMs of space S and time T with an arbitrary instruction set can be simulated by branching programs of size $2^{\mathcal{O}(S)}$ and time T. This underscores the importance of establishing lower bounds on time and space (and the tradeoff between them) in the branching program model. A tremendous amount of research has been devoted to this problem in the past two decades [Abr91, Ajt99, Ajt98, BM05, BJS01, BSS03, BST98, BV02, BW01, BC82, BRS83, Ger94, Juk95, Juk02, MNT93, Mor73, Oko91, Oko02, Pag01, Pip78, Pon98, Raz91, Win67, Yes84], and considerable progress has been made in proving lower bounds for less and less restricted branching-program models.

Loosely speaking, a *branching program* $\mathcal{B}$ is a finite directed acyclic graph, with a unique source node and one or more sink nodes. Each non-sink node is labeled by a variable and the edges out of the node correspond to the possible values of that variable. Executing the program on a given input corresponds to following a path from the source node to a sink node, using the values of the input variables to determine the edges to follow. We will give a precise definition of branching programs in this section.

B. Decision Branching programs

When dealing with *deterministic single-output* branching programs, the following is the model we will use throughout this paper. The definition below follows Beame, Saks, Sun, and Vee [BSS03].

Definition II.1 *Let $\mathcal{D}$ be a finite set of size q, and let $\mathcal{I}$ be a finite subset of $\mathbb{Z}$ of size n. A **q-way deterministic decision branching program** $\mathcal{B}$ on domain $\mathcal{D}$ with index set $\mathcal{I}$ is a graph $\mathcal{G}$ with the following properties:*

- ***a.*** *$\mathcal{G}$ is a finite edge-labeled and vertex-labeled directed acyclic graph, with a unique source node;*
- ***b.*** *There are two sink nodes in $\mathcal{G}$, one is labeled by 0 and the other by 1;*
- ***c.*** *Each non-sink node v of $\mathcal{G}$ is labeled by an index $i(v) \in \mathcal{I}$;*
- ***d.*** *The sink nodes have out-degree zero, all other nodes have out-degree q. For each non-sink node v, the set of edges starting at v is labeled by the elements of $\mathcal{D}$ so that all the q edges are labeled distinctly.*

For simplicity, we henceforth assume without loss of generality that the index set $\mathcal{I}$ in Definition II.1 is given by $\mathcal{I} = [n]$ and write $\mathcal{D}^{\mathcal{I}}$ as $\mathcal{D}^n$.

A ***computation*** by a decision branching program $\mathcal{B}$ on an input $\boldsymbol{x} \in \mathcal{D}^n$ starts at the source node s, reading the value of the variable $x_{i(s)}$ and following the edge labeled by that value. The process continues until one reaches a sink node. We say that $\mathcal{B}$ ***accepts an input $\boldsymbol{x}$*** if the sink node is labeled with a "1". Otherwise, we say that $\mathcal{B}$ ***rejects the input $\boldsymbol{x}$***. We let $\mathcal{B}^{-1}(1)$ denote the set of all inputs that $\mathcal{B}$ accepts. Thus $\mathcal{B}$ computes the function $f : \mathcal{D}^n \to \{0,1\}$ defined by $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} \in \mathcal{B}^{-1}(1)$.

C. Some Branching Program Terminology

The total number of nodes in $\mathcal{B}$ is called its ***size*** and denoted by $|\mathcal{B}|$. The ***space*** of $\mathcal{B}$ is then defined by $S = \log_q |\mathcal{B}|$. The length of a computation in $\mathcal{B}$ is the number of edges in the corresponding path. The ***time*** T of $\mathcal{B}$ (sometimes also called the ***length*** of $\mathcal{B}$) is defined as the maximum length of a computation in $\mathcal{B}$. Note that not all paths from the source to the sink in $\mathcal{G}$ are (valid) computations. We define the ***depth*** of $\mathcal{B}$ as the number of edges on the longest path from the source to the sink in $\mathcal{G}$.

A branching program $\mathcal{B}$ is said to be ***leveled*** if the set of nodes of $\mathcal{G}$ can be partitioned into an ordered collection of subsets $V_0, V_1, \ldots, V_\ell$, called ***levels***, such that the edges of $\mathcal{G}$ are always directed from a node at level V_{i-1} to a node at level V_i, for some $i \in \{1, 2, \ldots, \ell\}$. Pippenger [Pip78] proved that any branching program can be made leveled without affecting its time T while adding only $\mathcal{O}(\log T)$ to its space S. The ***width*** of such a leveled branching program is defined as $\max\{|V_0|, |V_1|, \ldots, |V_\ell|\}$.

A branching program $\mathcal{B}$ is said to be ***oblivious*** if the input variables are read in the same order along all possible paths from the source to the sink in $\mathcal{G}$. A ***read-r*** branching program is one in which each input variable is read at most r times along any path from the source to a sink in $\mathcal{G}$. A q-way branching program with $q = 2$ is said to be ***boolean***. A useful visualization of a boolean branching program is as a RAM with 1-bit-wide input registers and a working memory of S bits [Ajt99, BC82]. As already mentioned in the introduction, proving lower bounds on the time-space tradeoff is generally considered the most challenging for boolean branching programs. Indeed, as shown in [Pag01, Proposition 1], a factor of $\log_2 q$ is lost when converting any space or time lower bound from a q-way branching program to a boolean branching program.

The ***expected-time*** $\overline{T}$ of $\mathcal{B}$ is the mean time of a computation when the input $\boldsymbol{x}$ is chosen uniformly at random from $\mathcal{D}^n$. The nodes of $\mathcal{G}$ can be labeled with the integers $0, 1, \ldots, |\mathcal{B}|-1$ in $|\mathcal{B}|!$ different ways. Fix one such labeling. Then, for each input $\boldsymbol{x} \in \mathcal{D}^n$, the ***workspace*** required by $\mathcal{B}$ on input $\boldsymbol{x}$ is defined as the logarithm of the largest integer which occurs as a label of a node in the computation path. The ***expected-workspace*** is the mean workspace obtained when $\boldsymbol{x}$ is uniformly random over $\mathcal{D}^n$. The ***expected-space*** $\overline{S}$ of $\mathcal{B}$ can be now defined as the minimum of the expected-workspace, taken over all labellings.

The candidate functions for deriving time-space tradeoffs for branching programs will be from coding theory – these functions will be defined in Section II-D.

D. Error-Correcting Codes

Let F_q denote the finite field of order q. An error-correcting ***code*** C of length n over F_q is simply a subset of F_q^n. A ***linear code*** of dimension k is a k-dimensional subspace of F_q^n. If C is a linear code, then $|\mathrm{C}| = q^k$ for an integer k. We shall assume that $|\mathrm{C}| = q^k$ throughout, whether C is linear or not. An ***encoder*** for C is a one-to-one and onto (bijective) function $E_{\mathrm{C}} : \mathrm{F}_q^k \to \mathrm{C}$. The Hamming ***distance*** between two vectors in F_q^n is simply the number of positions where they differ. The ***minimum distance*** of a code C is the minimum Hamming distance between any two distinct vectors in C. When we say that C is ***an*** $(n,k,d)_q$ ***code***, we mean that $\mathrm{C} \subseteq \mathrm{F}_q^n$ is such that $|\mathrm{C}| = q^k$ and the minimum distance of C is at least d. If the order of the underlying field is either clear from the context or is not relevant, we will write (n,k,d) instead of $(n,k,d)_q$.

The ***rate*** of an (n,k,d) code is k/n and its ***relative distance*** is d/n. A set of codes characterized by a certain property is called a ***family of codes*** (e.g., the family of linear codes). A family of codes $\mathcal{F}$ is said to be ***asymptotically good*** if it contains an infinite sequence of codes ${}^1\mathrm{C}_q, {}^2\mathrm{C}_q, \ldots$ of increasing length n, such that both the rate and the relative distance of all the codes in this sequence are bounded away from zero as $n \to \infty$.

Let C be an (n,k,d) linear code over F_q. A ***generator matrix*** for C is any $k \times n$ matrix whose rows form a basis for C over F_q. The ***dual code*** $\mathrm{C}^\perp$ is the set of all $\boldsymbol{x} \in \mathrm{F}_q^n$ such that $\langle \boldsymbol{x}, \boldsymbol{c} \rangle = 0$ for all $\boldsymbol{c} \in \mathrm{C}$, where $\langle \cdot, \cdot \rangle$ is the usual inner product over F_q. Clearly, $\mathrm{C}^\perp$ is an $(n, n-k, d^\perp)$ linear code. A generator matrix for $\mathrm{C}^\perp$ is said to be a ***parity-check matrix*** for C.

An (n,k,d) code C is said to be ***maximum distance separable (MDS)*** if its minimum distance satisfies $d = n - k + 1$. The well-known Singleton bound implies that this minimum distance is the largest possible for an (n,k,d) code. For a comprehensive overview of MDS codes and their properties, see [MS77, Chapter 11].

Although we will be interested in decision functions related to error-correcting codes in this paper, it is useful to consider a few multiple-output functions related to coding. One such function is the encoder function E_{C} defined above. Another multiple-output function of importance is the *coset-identifier function*, which gives the coset (with respect to the code, C) to which an input $\boldsymbol{x} \in \mathrm{F}_q^n$ belongs. For a linear code this is equivalent to the *syndrome-vector function*:

Definition II.2 *If C is an (n,k,d) linear code over F_q, with generator matrix G and parity-check matrix H, we define the **syndrome function** $f_{\mathrm{C}}\colon \mathrm{F}_q^n \to \mathrm{F}_q^{n-k}$ and **dual syndrome function** $f_{\mathrm{C}}^{\perp}\colon \mathrm{F}_q^n \to \mathrm{F}_q^k$ as follows: $f_{\mathrm{C}}(\boldsymbol{x}) = H\boldsymbol{x}^t$ and $f_{\mathrm{C}}^{\perp}(\boldsymbol{x}) = G\boldsymbol{x}^t$.*

Time-space tradeoff results are known [BM05, SV06, San06] for the functions E_{C} and $f_{\mathrm{C}}^{\perp}$ defined above on the multiple-output branching program model. These results have been tabulated in Table I. In the next section we define and carefully study a closely related decision function.

III. A Decision Problem from Coding Theory

We will be interested in decision functions related to error-correcting codes in this paper. The *characteristic function* (also known as the *membership function*), defined below is a natural choice for a decision function:

Definition III.1 *Let C be code of length n over F_q. Its **characteristic function** is the function $\chi_{\mathrm{C}}\colon \mathrm{F}_q^n \to \{0,1\}$ defined by $\chi_{\mathrm{C}}(\boldsymbol{x}) = 1$ if and only if $\boldsymbol{x} \in \mathrm{C}$.*

However in our study of decision branching programs we found that another function related to the membership function of $\mathrm{C}^{\perp}$ is more convenient:

Definition III.2 *Let C be an (n,k,d) linear code over F_q. Let G be a generator matrix of this code. Let γ be a real constant such that $0 < \gamma < 1$. For an input $(\boldsymbol{x},\boldsymbol{y}) \in \mathrm{F}_q^n \times \mathrm{F}_q^k$ the decision function $f_{\mathrm{C},\gamma} : \mathrm{F}_q^n \times \mathrm{F}_q^k \to \{0,1\}$ evaluates true, namely $f_{\mathrm{C},\gamma}(\boldsymbol{x},\boldsymbol{y}) = 1$ if and only if at least a fraction γ of the k equations $G\boldsymbol{x}^t = \boldsymbol{y}^t$ are satisfied.*

For decision problems with large time-space tradeoffs, we observe the following general traits. First, the problem should have a nice algebraic structure for establishing the tradeoff. Second, the fraction of the acceptance set should preferably not decrease exponentially with the problem size. This second condition is necessary to obtain non-trivial tradeoffs. Both of these conditions can be conflicting and difficult to simultaneously ensure.

The decision function $f_{\mathrm{C},\gamma}$ defined as above appears in some form in most decoding problems encountered in coding theory. We have,

Lemma III.1 *Let $(\boldsymbol{x},\boldsymbol{y}) \in \mathrm{F}_q^n \times \mathrm{F}_q^k$ be an arbitrary tuple of vectors. If $\mathcal{N} \subseteq [k]$ then we denote by $G_{\mathcal{N}}$ the sub-matrix of G consisting of only those rows which are indexed by $\mathcal{N}$. Then*

$$(\boldsymbol{x},\boldsymbol{y}) \in f_{\mathrm{C},\gamma}^{-1}(1) \iff \exists\, \mathcal{N} \subseteq [k] \text{ such that } |\mathcal{N}| \geqslant \lceil \gamma k \rceil \text{ and } G_{\mathcal{N}}\boldsymbol{x}^t - I_{\mathcal{N}}\boldsymbol{y}^t = \boldsymbol{0}^t$$

and,

$$\begin{aligned}(\boldsymbol{x},\boldsymbol{y}) \in f_{\mathrm{C},\gamma}^{-1}(0) \iff &\exists\, \mathcal{N} \subseteq [k] \text{ and } \exists\, \boldsymbol{z} \in \{\mathrm{F}_q \backslash \{0\}\}^{|\mathcal{N}|} \\ &\text{such that } |\mathcal{N}| > \lfloor (1-\gamma)k \rfloor \text{ and } G_{\mathcal{N}}\boldsymbol{x}^t - I_{\mathcal{N}}\boldsymbol{y}^t = \boldsymbol{z}^t\end{aligned}$$

where $I_{\mathcal{N}}$ consists of the rows of the identity matrix I indexed by $\mathcal{N}$.

Proof. Follows directly from the definition of $f_{\mathrm{C},\gamma}$. ■

Lemma III.2 *The acceptance ratio $\alpha \stackrel{def}{=} \frac{|f_{\mathrm{C},\gamma}^{-1}(1)|}{q^{n+k}}$ is given by $\alpha = \frac{V_q(k,\lfloor(1-\gamma)k\rfloor)}{q^k}$, where $V_q(N,r)$ is the volume of an N-dimensional q-ary Hamming ball $B_q(N,r)$ of radius r. The asymptotic acceptance ratio is given by*

(i) $\lim_{k\to\infty} \frac{\log_q \alpha}{k} = H_q(1-\gamma) - 1, \qquad if\ \frac{1}{q} < \gamma \leqslant 1,$

(ii) $\lim_{k\to\infty} \frac{\log_q (1-\alpha)}{k} = H_q(1-\gamma) - 1, \qquad else,$

where $H_q(\cdot)$ denotes the q-ary entropy function.

Proof. Consider an input $(\boldsymbol{x},\boldsymbol{y}) \in \mathrm{F}_q^n \times \mathrm{F}_q^k$. By Lemma III.1, it is accepted iff $\exists$ an index set $\mathcal{N} \subset [k]$ such that $|\mathcal{N}| \geqslant \lceil \gamma k \rceil$ and $G_{\mathcal{N}}\boldsymbol{x}^t - I_{\mathcal{N}}\boldsymbol{y}^t = \boldsymbol{0}^t$. This happens when $d_H(\boldsymbol{y}^t, G_{\mathcal{N}}\boldsymbol{x}^t) \leqslant \lfloor (1-\gamma)k \rfloor$, where $d_H(.,.)$ denotes the Hamming distance between two q-ary vectors. Since there are a total of q^n such $\boldsymbol{x}$ we have $|f_{\mathrm{C},\gamma}^{-1}(1)| = q^n V_q(k, \lfloor (1-\gamma)k \rfloor)$ and the result follows.

It can be shown that $\binom{k}{i}(q-1)^i < \binom{k}{i+1}(q-1)^{(i+1)}$ as long as $i < \frac{(q-1)k}{q} - \frac{1}{q}$; beyond this, the inequality gets reversed.

(i) Here $\frac{1}{q} < \gamma \leqslant 1$. Therefore $\alpha q^k = \sum_{i=0}^{\lfloor (1-\gamma)k \rfloor} \binom{k}{i}(q-1)^i$ can be bounded as $\binom{k}{i_0}(q-1)^{i_0} \leqslant \alpha q^k \leqslant (1+i_0)\binom{k}{i_0}(q-1)^{i_0}$, where $i_0 = \lfloor (1-\gamma)k \rfloor$. Now take logarithms and then the limit as $k \to \infty$.

(ii) In this case, $0 \leqslant \gamma \leqslant \frac{1}{q}$. Therefore $(1-\alpha)q^k = \sum_{i=\lfloor (1-\gamma)k \rfloor + 1}^{k} \binom{k}{i}(q-1)^i$ can be bounded as $\binom{k}{i_0}(q-1)^{i_0} \leqslant (1-\alpha)q^k \leqslant (k-i_0+1)\binom{k}{i_0}(q-1)^{i_0}$, where $i_0 = \lfloor (1-\gamma)k \rfloor + 1$. Taking logarithms and limit the result follows. ■

It is easy to see that depending on γ, either α or $(1-\alpha)$ can decrease exponentially with k. For larger alphabets most often α is exponentially small. The decision function $f_{\mathrm{C},\gamma}$ has the interesting property that both the acceptance and rejection sets have a nice linear algebraic structure, which we intend to take advantage of.

In prior work, several quadratic time-space bounds valid for $T = \mathcal{O}(n^2)$ for multiple-output branching programs where successfully derived. In all those instances, the principal means for obtaining these bounds was furnished by a convenient measure of the number of correct outputs generated by the program on any partial computation path. In single output branching programs, there is only one output variable precluding use of this particular strategy. For example, in the papers of Beame et al [BSS03] and Ajtai [Ajt99, Ajt98] the properties of a `core` set of input variables read exclusively by a partial computation path is used to obtain the time-space tradeoffs. However this technique has so far not resulted in a quadratic time-space tradeoff for Boolean decision programs which is valid for any T. In this paper, we use a new measure of progress which is made possible by a careful choice of the decision function. Our measure of progress in computing the decision function is the number of variables corresponding to the vector $\boldsymbol{y}$ that $\mathcal{B}$ reads during the computation and which in turn must satisfy certain linear constraints inherent in the problem.

IV. A Quadratic Time-Space Tradeoff for $f_{\mathrm{C},\gamma}$

Let $\mathcal{B}$ be a q-ary deterministic decision branching program which computes $f_{\mathrm{C},\gamma}$. Let us assign a labeling to the nodes of $\mathcal{B}$ which minimizes the expected-space of $\mathcal{B}$.

Definition IV.1 *A **branching sub-program** $\mathscr{P}$ of a branching program $\mathcal{B}$ is a collection of partial computational paths in $\mathcal{B}$. Let P be an arbitrary partial computational path in the collection $\mathscr{P}$. P consists of a partial computational path which starts from a node in $\mathcal{B}$ and ends in another node in $\mathcal{B}$. Each such member of $\mathscr{P}$ can have a different start node (and/or sink node). The expected time of a branching sub-program is the average length of the partial computations in the sub-program. The expected space of $\mathscr{P}$ is the average over all partial computations P in $\mathscr{P}$ of the maximum node labeling on any node in P. The node labeling is assumed to be assigned to the nodes in $\mathcal{B}$ so as to minimize the expected space of $\mathcal{B}$.*

Now let $\mathcal{B}_1$ be the branching sub-program consisting of all computation paths starting from the root node of $\mathcal{B}$, and ending in the accepting sink node. Similarly, let $\mathcal{B}_0$ be the branching sub-program consisting

of all computation paths from the root node of $\mathcal{B}$ and ending in the rejecting sink node. The following relations are obvious:

Lemma IV.1 *Let the expected-time and expected-space of $\mathcal{B}$ be $\overline{T}$ and $\overline{S}$. Similarly, let the expected-time and expected-space of the computational paths in $\mathcal{B}_0$ and $\mathcal{B}_1$ be denoted as $\overline{T}_0$, $\overline{S}_0$ and $\overline{T}_1$, $\overline{S}_1$ respectively. Then,*

$$\overline{T} = \alpha\overline{T}_1 + (1-\alpha)\overline{T}_0 \quad (1)$$
$$\overline{S} = \alpha\overline{S}_1 + (1-\alpha)\overline{S}_0 \quad (2)$$

where $\alpha \stackrel{def}{=} \frac{|\mathcal{B}^{-1}(1)|}{q^{n+k}}$ is the acceptance ratio.

Proof. Let us denote the computation time on input $(\boldsymbol{x},\boldsymbol{y})$ by $t_{(x,y)}$. Then

$$\overline{T} = \sum_{(\boldsymbol{x},\boldsymbol{y})} \frac{t_{(x,y)}}{q^{n+k}} = \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}^{-1}(1)} \frac{t_{(x,y)}}{q^{n+k}} + \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}^{-1}(0)} \frac{t_{(x,y)}}{q^{n+k}} = \alpha\overline{T}_1 + (1-\alpha)\overline{T}_0$$

since $\mathcal{B}^{-1}(1)$ and $\mathcal{B}^{-1}(0)$ disjointly partition $\mathrm{F}_q^n \times \mathrm{F}_q^k$. Expected space is treated in a similar manner. ■

We now proceed to obtain a time-space tradeoff valid for $\mathcal{B}$. Towards this, we will first obtain time-space tradeoffs for $\mathcal{B}_1$ and then $\mathcal{B}_0$ separately, before using Lemma IV.1 to obtain the desired result. We have,

Lemma IV.2 *Let $\mathcal{B}$ be a q-way branching program of depth δ. For $r = 1, 2, \ldots, \delta$, let η_r denote the number of computation paths in $\mathcal{B}$ which read exactly r different input variables. Then*

$$\sum_{r=1}^{\delta} \eta_r q^{-r} = 1 \quad (3)$$

Proof. Let P denote a specific computation path in $\mathcal{B}$ that reads exactly r input variables, and assume w.l.o.g. that these variables are $y_1, y_2, \ldots, y_r$. Since P is a *computation path*, the labels of the edges of P must be consistent — that is, if v and v' are different nodes of P that read the same variable, then the edges of P starting at v and v' must have the same label. Thus let $u_1, u_2, \ldots, u_r \in \mathcal{D}$ denote the labels on the edges of P that correspond[1] to $y_1, y_2, \ldots, y_r$, respectively. Then, the computation of $\mathcal{B}$ upon input $\boldsymbol{x} \in \mathcal{D}^n$ follows the path P from the source to the sink iff $x_1 = u_1, x_2 = u_2, \cdots, x_r = u_r$. Now, suppose that $\boldsymbol{x}$ is chosen uniformly at random from $\mathcal{D}^n$. Then, by the foregoing discussion, the probability that the computation of $\mathcal{B}$ on input $\boldsymbol{x}$ follows the path P is q^{-r}. Let $\mathscr{P}$ denote the set of *all* computation paths in $\mathcal{B}$. Then

$$1 = \sum_{P \in \mathscr{P}} \Pr\{\text{computation on } \boldsymbol{x} \text{ follows } P\} = \sum_{r=1}^{\delta} \eta_r q^{-r} \quad (4)$$

since for every $\boldsymbol{x} \in \mathcal{D}^n$, the computation of $\mathcal{B}$ upon input $\boldsymbol{x}$ follows *exactly one* path in $\mathscr{P}$. Thus we are summing over probabilities of disjoint events that partition the sample space. ■

Lemma IV.2 leads to the following elementary result:

Lemma IV.3 *Let $\mathcal{B}$ be a q-way branching program of depth δ. Let $\mathscr{P}$ be a branching sub-program from $\mathcal{B}$. For $r = 1, 2, \ldots, \delta$, let η_r denote the number of computation paths in $\mathscr{P}$ which read exactly r different input variables. Then*

$$\sum_{r=1}^{\delta} \eta_r q^{-r} \leqslant 1 \quad (5)$$

[1] Henceforth, we shall say that $u_1, u_2, \ldots, u_r$ are the values which the path P ***enforces*** on the variables $y_1, y_2, \ldots, y_r$.

Proof. Follows immediately from Lemma IV.2. Not all terms in the summation of probabilities in (4) are present. Each term is non-negative, implying the inequality. ■

We will make use of the following property:

Lemma IV.4 *Let G be a $k \times n$ generator matrix for an (n,k,d) linear code C. Let a,b be positive integers with $a \leqslant d-1$ and $b \leqslant k$. Then every $b \times (n-a)$ sub-matrix G' of G is full-rank.*

Proof. Since $d-1 \leqslant n-k$ for any (n,k,d) code by the Singleton bound, we observe that $b \leqslant n-a$. Thus what we need to show is that $\operatorname{rank} G' = b$. To this end, let G'' be the $k \times (n-a)$ column sub-matrix of G such that G' is a row sub-matrix of G''. It is well known that for $a \leqslant d-1$, every $n-a$ columns of G contain an information set. Hence $\operatorname{rank} G'' = k$ and its k rows are linearly independent. It follows that the b rows of G' are also independent, and so $\operatorname{rank} G' = b$. ■

The following definition which fixes the notion of *partial decision* will also be used extensively:

Definition IV.2 *Let $\mathcal{B}$ be a branching program, and let $\mathscr{P}$ be a branching sub-program from $\mathcal{B}$. Let $c \leqslant k$ be a positive integer. Consider a specific computation path P in $\mathscr{P}$ and a specific vector $(\boldsymbol{x},\boldsymbol{y}) \in \mathrm{F}_q^n \times \mathrm{F}_q^k$. We say that P $\langle c \rangle$-**decides** $(\boldsymbol{x},\boldsymbol{y})$ **with respect to** $f_{\mathrm{C},\gamma}$ iff*

(i) The set of all nodes and edges in the path P is a subset of the set of nodes and edges in the computational path $P'_{(\boldsymbol{x},\boldsymbol{y})}$ in $\mathcal{B}$ corresponding to $(\boldsymbol{x},\boldsymbol{y})$, so that the labels of all the edges in P are consistent with $(\boldsymbol{x},\boldsymbol{y})$.

(ii) The path P reads at least c of the input variables corresponding to the positions in $\boldsymbol{y}$.

(iii) There exists an index set $\mathcal{N} = \mathcal{N}_{(\boldsymbol{x},\boldsymbol{y})} \subseteq [k]$ with $|\mathcal{N}_{(\boldsymbol{x},\boldsymbol{y})}| \geqslant c$ such that,

(a) If $(\boldsymbol{x},\boldsymbol{y}) \in f_{\mathrm{C},\gamma}^{-1}(1)$; then $G_{\mathcal{N}}\boldsymbol{x}^t - I_{\mathcal{N}}\boldsymbol{y}^t = \boldsymbol{0}^t$

(b) If $(\boldsymbol{x},\boldsymbol{y}) \in f_{\mathrm{C},\gamma}^{-1}(0)$; then $G_{\mathcal{N}}\boldsymbol{x}^t - I_{\mathcal{N}}\boldsymbol{y}^t = \boldsymbol{z}^t$ where $\boldsymbol{z} \in \{\mathrm{F}_q \backslash \{0\}\}^{|\mathcal{N}|}$.

*We say that $\mathscr{P}$ $\langle c \rangle$-**decides** $(\boldsymbol{x},\boldsymbol{y})$ **with respect to** $f_{\mathrm{C},\gamma}$ iff the computation path that $\mathscr{P}$ follows upon input $(\boldsymbol{x},\boldsymbol{y})$ (if such a path exists) $\langle c \rangle$-decides $\boldsymbol{x}$ w.r.t. $f_{\mathrm{C},\gamma}$.*

Let us consider the branching sub-programs $\mathcal{B}_1$ and $\mathcal{B}_0$ obtained from $\mathcal{B}$. By the standard technique, level both $\mathcal{B}_1$ and $\mathcal{B}_0$ separately. To make $\mathcal{B}_j$, $j \in \{0,1\}$ leveled, first add q edges labeled by all the elements of $\mathcal{D} = \mathrm{F}_q$ from the sink node of $\mathcal{B}_j$ to itself. Then, replicate the resulting graph $\delta+1$ times, where δ is the depth of $\mathcal{B}_j$. Such replication produces the $\delta+1$ *levels* $V_0, V_1, \ldots, V_\delta$, with $|V_i| = |\mathcal{B}_j|$ for all i. Now, for $i = 0,1,\ldots,\delta-1$, redirect all edges from nodes at level V_i to nodes at level V_{i+1}. Finally, delete all nodes that are unreachable from the source node at level V_0 (***deleting a node*** means also deleting all the edges that are incident upon this node). This produces a leveled set of computation paths $\mathcal{B}_j$ with $\delta+1$ levels, whose source φ is the source node at level V_0 (in fact $V_0 = \{\varphi\}$) and whose sink ϕ is the sink node at level V_δ (in fact $V_\delta = \{\phi\}$).

Next, we truncate each $\mathcal{B}_j$, $j \in \{0,1\}$ to a depth of $\lceil (1+\tau)\overline{T}_j \rceil$, where $\tau > 0$ is an absolute constant to be fixed later. Here, ***truncation to depth*** T means deleting all the nodes that are unreachable from the sink node at level V_T when the direction of all edges is reversed. This produces a leveled branching sub-program with $T+1$ levels, which we denote by $\mathcal{B}'_j$. Observe that $\mathcal{B}_j$ and $\mathcal{B}'_j$ compute the same function, while the time and space of $\mathcal{B}'_j$ are given by $T'_j = T_j$ and $S'_j \leqslant S_j + \log T_j$.

We call these two branching sub-programs $\mathcal{B}'_j$, $j \in \{0,1\}$. Let $(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}^{-1}(j)$ for $j \in \{0,1\}$ and let $t_{(x,y)}$ denote the computation time on input $(\boldsymbol{x},\boldsymbol{y})$ of $\mathcal{B}$. Then by a simple application of Markov inequality,

$$\Pr\{t_{(x,y)} \leqslant \lceil (1+\tau)\overline{T}_j \rceil\} \geqslant 1 - \frac{\overline{T}}{\lceil (1+\tau)\overline{T} \rceil} \geqslant \frac{\tau}{1+\tau}$$

Hence the branching sub-programs $\mathcal{B}'_j$, $j \in \{0,1\}$ successfully complete calculating the decision function with probability at least $\frac{\tau}{1+\tau}$ when $(\boldsymbol{x},\boldsymbol{y})$ is chosen uniformly at random from $f^{-1}_{\mathrm{C},\gamma}(j)$, $j \in \{0,1\}$. The expected-space $\overline{S}'_j$ required for $\mathcal{B}'_j$ cannot exceed $\overline{S}_j + \log_2 \lceil (1+\tau)\overline{T}_j \rceil$ for $j \in \{0,1\}$. The expected-time $\overline{T}'_j$ of $\mathcal{B}'_j$ increases due to the leveling process and decreases during truncation. We will not need an estimate of $\overline{T}'_j$ in this paper. The following result provides us with a crucial measure of progress of computation.

Lemma IV.5 *Let $\mathcal{B}'_j$, $j \in \{0,1\}$ be segmented into B blocks. Then*

(i) Let $(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}'^{-1}_1(1)$. There exists some block $b_{(x,y)}$ and a branching sub-program fully contained in that block which $\langle \lceil \frac{\lceil \gamma k \rceil}{B} \rceil \rangle$-decides $(\boldsymbol{x},\boldsymbol{y})$ w.r.t. $f_{\mathrm{C},\gamma}$.

(ii) Let $(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}'^{-1}_0(0)$. There exists some block $b_{(x,y)}$ and a branching sub-program fully contained in that block which $\langle \lceil \frac{(\lfloor (1-\gamma)k \rfloor + 1)}{B} \rceil \rangle$-decides $(\boldsymbol{x},\boldsymbol{y})$ w.r.t. $f_{\mathrm{C},\gamma}$.

Proof. First let us observe that due to the truncation process, there are some inputs which do not reach a sink node in either of the two branching sub-programs $\mathcal{B}'_1$ or $\mathcal{B}'_0$. Therefore in general $\mathcal{B}'^{-1}_1(1) \cup \mathcal{B}'^{-1}_0(0)$ is not equal to $\mathrm{F}_q^n \times \mathrm{F}_q^k$.

(i) By assumption, $(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}'^{-1}_1(1)$. Therefore, the computation path P which $\mathcal{B}'_1$ follows on $(\boldsymbol{x},\boldsymbol{y})$ ends in the accepting sink node. Now assume to the contrary that there are no blocks which $\langle \lceil \frac{\lceil \gamma k \rceil}{B} \rceil \rangle$-decide $(\boldsymbol{x},\boldsymbol{y})$ w.r.t. $f_{\mathrm{C},\gamma}$. By hypothesis, the condition (i) is satisfied in Definition IV.2. Therefore conditions (ii) and/or $(iii)(a)$ must be the ones which are not satisfied. That is, either there is no block in $\mathcal{B}'_1$ which reads more than $\lceil \frac{\lceil \gamma k \rceil}{B} \rceil$ of the variables corresponding to the input vector $\boldsymbol{y}$, and/or the read variables do not satisfy the linear constraints of $(iii)(a)$ in Definition IV.2. Therefore the computational path P that $\mathcal{B}$ follows on input $(\boldsymbol{x},\boldsymbol{y})$ does not read more than $\lceil \gamma k \rceil - 1$ of the variables corresponding to $\boldsymbol{y}$ which simultaneously also satisfy the linear constraints in Lemma III.1. This means that there are always some inputs which are erroneously accepted by $\mathcal{B}$ due to this computational path P. These are inputs of the form $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$, such that $(\boldsymbol{x},\boldsymbol{y})$ and $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$ coincide on the variables actually read by P and yet $\boldsymbol{g}_i \cdot \tilde{\boldsymbol{x}} = \tilde{y}_i$ is satisfied for fewer than $\lceil \gamma k \rceil$ rows of G. This is not possible for a deterministic decision branching program $\mathcal{B}$ which should be correct on all possible inputs. This proves the first claim.
(ii) This part of the proof uses a similar technique as the first part. Once again if we assume to the contrary, condition (ii) and/or $(iii)(b)$ of Definition IV.2 must be the ones which are not satisfied, and this cannot be the case for a deterministic program. ■

We have the following result:

Lemma IV.6 *Let G be the generator matrix of an (n,k,d) linear code C over F_q. Let $0 < \gamma < 1$ be an absolute real constant and let $c \leqslant k$ be a positive integer. Let the acceptance ratio of the decision function $f_{\mathrm{C},\gamma}$ be denoted as α. Let $\mathscr{P}$ be a branching sub-program from a branching program $\mathcal{B}$ of depth $\delta < d + c$.*

(i) If $(\boldsymbol{x},\boldsymbol{y})$ is chosen uniformly at random from $f^{-1}_{\mathrm{C},\gamma}(1)$, then

$$\Pr\{\mathscr{P} \ \langle c \rangle\text{-decides } (\boldsymbol{x},\boldsymbol{y}) \text{ w.r.t. } f_{\mathrm{C},\gamma}\} \leqslant \frac{(\frac{1}{q})^c}{\alpha}$$

(ii) If $(\boldsymbol{x},\boldsymbol{y})$ is chosen uniformly at random from $f^{-1}_{\mathrm{C},\gamma}(0)$, then

$$\Pr\{\mathscr{P} \ \langle c \rangle\text{-decides } (\boldsymbol{x},\boldsymbol{y}) \text{ w.r.t. } f_{\mathrm{C},\gamma}\} \leqslant \frac{(1-\frac{1}{q})^c}{1-\alpha}$$

Proof. In order not to repeat parts of the proof, we initially consider any $j \in \{0,1\}$ and then specialize for the two particular cases when necessary. Let $\mathcal{E}$ be the event that the branching program $\mathcal{B}$ $\langle c \rangle$-decides

$(\boldsymbol{x},\boldsymbol{y})$ w.r.t. $f_{\mathrm{C},\gamma}$, $j\in\{0,1\}$. For each computation path P in $\mathcal{B}$, let $\mathcal{E}_P$ be the event that P $\langle c\rangle$-decides $(\boldsymbol{x},\boldsymbol{y})$ w.r.t. $f_{\mathrm{C},\gamma}$. Then clearly

$$\Pr\{\mathcal{E}\}=\sum_{P\in\mathscr{P}}\Pr\{\mathcal{E}_P\} \tag{6}$$

Now let P be a specific (fixed) computation path in $\mathcal{B}$ and suppose that

(a) P $\langle c\rangle$-decides $(\boldsymbol{x},\boldsymbol{y})$ w.r.t. $f_{\mathrm{C},\gamma}$, and
(b) P reads exactly r input variables.

The assumption that P $\langle c\rangle$-decides $(\boldsymbol{x},\boldsymbol{y})$ in particular implies that P is contained in the computation path that $\mathcal{B}$ will follow upon input $(\boldsymbol{x},\boldsymbol{y})$. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be the vector of values that P enforces on the input variables corresponding to $\boldsymbol{x}$ and $\boldsymbol{y}$ that it reads respectively. Then $(\boldsymbol{x},\boldsymbol{y})$ must satisfy $x_j=u_j$ if x_j is read by P and $y_j=v_j$ if y_j is read by P. All events are considered subject to the condition that $(\boldsymbol{x},\boldsymbol{y})\in f_{\mathrm{C},\gamma}^{-1}(j)$.

Therefore by Definition IV.2 and Lemma III.1 we have the following:

(i) $j=1$: P induces a set of $b\geqslant c$ linear equations of the form $\sum_{l=1}^{n}g_{i,l}x_l-y_i=0$ which $(\boldsymbol{x},\boldsymbol{y})$ must satisfy. This also implies that $r\geqslant b$. Together the induced system of linear equations is $G^*\boldsymbol{x}^t-I_b^*\boldsymbol{y}^t=\boldsymbol{0}^t$. Here G^* consists of b distinct rows of the generator matrix G with a possible permutation of the columns to match the variable ordering in $\boldsymbol{x}$ and I_b^* consists of the corresponding b rows of the $k\times k$ identity matrix I. $\boldsymbol{0}$ corresponds to the vector of correct decisions made by P. Combining all of the above, we find that $(\boldsymbol{x},\boldsymbol{y})$ must satisfy the following system of $r+b$ linear equations:

$$\left[\begin{array}{c|c}I_{r-b}^* & \\ \hline \boldsymbol{0}_{b\times n} & I_b^* \\ \hline G^* & -I_b^*\end{array}\right]_{(r+b)\times(n+k)}\left[\begin{array}{c}\boldsymbol{x}^t\\ \hline \boldsymbol{y}^t\end{array}\right]_{(n+k)\times 1}=\left[\begin{array}{c}\boldsymbol{u}^t\\ \hline \boldsymbol{v}^t\\ \hline \boldsymbol{0}^t\end{array}\right]_{(r+b)\times 1} \tag{7}$$

where I_{r-b}^* are some $(r-b)$ rows of the $(n+k)\times(n+k)$ identity matrix corresponding to the input variables read by P except the b input variables in $\boldsymbol{y}$ which belong to the set of satisfied linear constraints. Those input variables are counted as part of I_b^*. Let M be the $(r+b)\times(n+k)$ matrix in (7). Then $\operatorname{rank}M=\operatorname{rank}I_{r-b}^*+\operatorname{rank}I_b^*+\operatorname{rank}G'$, where G' is the $b\times(n-r+b)$ matrix consisting of some $(n-r+b)$ columns of G^* which are zero columns in I_{r-b}^*. Since $r\leqslant\delta<d+c$, we conclude that $\operatorname{rank}G'=b$ by Lemma IV.4 and $\operatorname{rank}M=r+b$. It follows that the linear system (7) has exactly $q^{n+k-b-r}$ distinct solutions in F_q^{n+k}, and so

$$\Pr\{\mathcal{E}_P\}=\frac{q^{n+k-b-r}}{|f_{\mathrm{C},\gamma}^{-1}(1)|}\leqslant\frac{q^{-(c+r)}}{\alpha} \tag{8}$$

Observe that the bound on $\Pr\{\mathcal{E}_P,(\boldsymbol{x},\boldsymbol{y})\in f_{\mathrm{C},\gamma}^{-1}(1)\}$ in (8) depends only on the number r of the input variables that P reads. If there are η_r computational paths in $\mathscr{P}$ which read exactly r variables then,

$$\Pr\{\mathcal{E}\}\leqslant\sum_{r=1}^{\delta}\eta_r q^{-r}\frac{q^{-c}}{\alpha}\leqslant\frac{q^{-c}}{\alpha}$$

where the first inequality follows from (6) and (8), while the second one follows from Lemma IV.3.

(ii) $j=0$: This case is similar. P induces a set of $b\geqslant c$ linear equations of the form $\sum_{l=1}^{n}g_{i,l}x_l-y_i=z_i$ which $(\boldsymbol{x},\boldsymbol{y})$ must satisfy, where $z_i\in\{\mathrm{F}_q\backslash\{0\}\}$. This again implies that $r\geqslant b$. Together the induced system of linear equations is $G^*\boldsymbol{x}^t-I_b^*\boldsymbol{y}^t=\boldsymbol{z}^t$. Here G^* consists of b distinct rows of the generator matrix G with a possible permutation of the columns to match the variable ordering in $\boldsymbol{x}$ and I_b^* consists of the corresponding b rows of the $k\times k$ identity matrix I. $\boldsymbol{z}$ corresponds to the c linear constraints satisfied

by P w.r.t. $f_{\mathrm{C},\gamma}$ and can be any vector from $\{\mathrm{F}_q\backslash\{0\}\}^b$. Therefore $(\boldsymbol{x},\boldsymbol{y})$ must satisfy the following system of $r+b$ linear equations:

$$\left[\begin{array}{c|c} I^*_{r-b} & \\ \hline \boldsymbol{0}_{b\times n} & I^*_b \\ \hline G^* & -I^*_b \end{array}\right]_{(r+b)\times(n+k)} \left[\begin{array}{c} \boldsymbol{x}^t \\ \hline \boldsymbol{y}^t \end{array}\right]_{(n+k)\times 1} = \left[\begin{array}{c} \boldsymbol{u}^t \\ \hline \boldsymbol{v}^t \\ \hline \boldsymbol{z}^t \end{array}\right]_{(r+b)\times 1} \tag{9}$$

where I^*_{r-b} are some $(r-b)$ rows of the $(n+k)\times(n+k)$ identity matrix corresponding to the input variables read by P except the b input variables in $\boldsymbol{y}$ which belong to the set of satisfied linear constraints. Those input variables are counted as part of I^*_b. Let M be the $(r+b)\times(n+k)$ matrix in (7). Then $\operatorname{rank} M = \operatorname{rank} I^*_{r-b} + \operatorname{rank} I^*_b + \operatorname{rank} G'$, where G' is the $b\times(n-r+b)$ matrix consisting of some $(n-r+b)$ columns of G^* which are zero columns in I^*_{r-b}. Since $r \leqslant \delta < d+c$, we conclude that $\operatorname{rank} G' = b$ by Lemma IV.4 and $\operatorname{rank} M = r+b$. It follows that the linear system (9) has has at most $(q-1)^b \cdot q^{n+k-b-r}$ distinct solutions in F_q^n, and so

$$\Pr\{\mathcal{E}_P\} \leqslant \frac{(q-1)^b \cdot q^{n+k-b-r}}{|f^{-1}_{\mathrm{C},\gamma}(0)|} \leqslant \frac{q^{-r}(1-\frac{1}{q})^c}{1-\alpha} \tag{10}$$

The bound on $\Pr\{\mathcal{E}_P, (\boldsymbol{x},\boldsymbol{y}) \in f^{-1}_{\mathrm{C},\gamma}(0)\}$ in (10) depends only on the number r of the input variables that P reads. If there are η_r computational paths in $\mathscr{P}$ which read exactly r variables then,

$$\Pr\{\mathcal{E}\} \leqslant \sum_{r=1}^{\delta} \eta_r q^{-r} \frac{(1-\frac{1}{q})^c}{1-\alpha} \leqslant \frac{(1-\frac{1}{q})^c}{1-\alpha}$$

where the first inequality follows from (6) and (10), while the second follows from Lemma IV.3. ■

Theorem IV.1 *Let C be an (n,k,d) linear code over F_q. Let $\mathcal{B}$ be a q-way branching program which computes the decision function $f_{\mathrm{C},\gamma}$ in expected-time $\overline{T}$ and expected-space $\overline{S}$. Then $\mathcal{B}$ satisfies the time-space tradeoff given by:*

$$6\overline{T}\left(\overline{S} + \log_2 \overline{T} + 7\right) \geqslant kd \cdot \max\left\{\alpha^2\gamma \log_2 q,\ \ (1-\alpha)^2(1-\gamma)\log_2 \tfrac{q}{q-1}\right\} \tag{11}$$

where $\alpha \overset{\mathrm{def}}{=} \frac{|\mathcal{B}^{-1}(1)|}{q^{n+k}}$ is the acceptance ratio for $\mathcal{B}$.

Proof. We obtain tradeoffs for $\mathcal{B}_j$, $j\in\{0,1\}$ separately and then use Lemma IV.1. So consider $\mathcal{B}_j$. Earlier we used a leveling and truncation procedure to obtain $\mathcal{B}'_j$. We also showed that with probability at least $\frac{\tau}{1+\tau}$, this leveled and truncated collection of computation paths (branching sub-programs) successfully completes calculating the decision function $f_{\mathrm{C},\gamma}(\boldsymbol{x},\boldsymbol{y})$ for an $(\boldsymbol{x},\boldsymbol{y})$ chosen uniformly at random from $f^{-1}_{\mathrm{C},\gamma}(j)$.

Let us now segment each $\mathcal{B}'_j$, $j\in\{0,1\}$ separately into B_j equal sized blocks of depth δ_j (except perhaps the last block, which could be shorter). For Lemma IV.6 to be applicable, we should ensure that $\delta_j < d+c_j$. Here c_j are given by an application of Lemma IV.5 thus:

(i) $j=1$: For every such $(\boldsymbol{x},\boldsymbol{y})$ for which the acceptance decision is successfully made by $\mathcal{B}'_1$, there must be some block which correctly decided in the affirmative a set of questions of the form

$$\boldsymbol{g}_i \cdot \boldsymbol{x} - y_i \overset{?}{=} 0 \tag{12}$$

for all $i\in\mathcal{N}_{(x,y)}$, where $\mathcal{N}_{(x,y)} \subseteq [k]$ is an index set such that $c_1 \overset{\mathrm{def}}{=} |\mathcal{N}_{(x,y)}| \geqslant \lceil \frac{\lceil \gamma k\rceil}{B_1}\rceil \geqslant 1$.

(ii) $\underline{j=0}$: Similarly, for every such $(\boldsymbol{x},\boldsymbol{y})$ for which the rejection decision is successfully made by $\mathcal{B}'_0$, there must be some block which correctly decided in the affirmative a set of questions of the form

$$\boldsymbol{g}_i \cdot \boldsymbol{x} - y_i \overset{?}{\neq} 0 \tag{13}$$

for all $i \in \mathcal{N}_{(x,y)}$, where $\mathcal{N}_{(x,y)} \subseteq [k]$ is an index set such that $c_0 \overset{def}{=} |\mathcal{N}_{(x,y)}| \geqslant \lceil \frac{(\lfloor (1-\gamma)k \rfloor + 1)}{B_0} \rceil \geqslant 1$.

We see that choosing $\delta_j = d$ satisfies the requirements and so $B_j = \left\lceil \frac{\lceil (1+\tau)\overline{T}_j \rceil}{d} \right\rceil$.

Let b be any of the blocks. Let v be an arbitrary node on the starting boundary level ℓ_b of this block. We will use the following notation:

$$p_{j,v}(\boldsymbol{x},\boldsymbol{y}) \overset{def}{=} \Pr\{\mathscr{P}_v \ \langle c_j \rangle\text{-decides } (\boldsymbol{x},\boldsymbol{y}) \text{ w.r.t. } f_{\mathrm{C},\gamma}\}$$

where $\mathscr{P}_v$ is the branching sub-program consisting of a collection of partial computation paths starting at node v up to but not including nodes in the next boundary level ℓ_{b+1}.

The probability of successful computation of the decision function $f_{\mathrm{C},\gamma}$ is upper bounded by the probability of the existence of a block with such an index set $\mathcal{N}_{(x,y)}$. For an $(\boldsymbol{x},\boldsymbol{y})$ chosen uniformly at random from $f^{-1}_{\mathrm{C},\gamma}(j)$ this later probability is at most

$$\sum_b \sum_{v \in \ell_b} p_{j,v}(\boldsymbol{x},\boldsymbol{y}) = \sum_b \sum_{\substack{v \in \ell_b: \\ l(v) < 2^{\sigma \overline{S}'_j}}} p_{j,v}(\boldsymbol{x},\boldsymbol{y}) + \sum_b \sum_{\substack{v \in \ell_b: \\ l(v) \geqslant 2^{\sigma \overline{S}'_j}}} p_{j,v}(\boldsymbol{x},\boldsymbol{y})$$

where $l(v)$ is the optimal labeling earlier assigned to the node v and $\sigma > 1$ is an absolute constant to be fixed later. By Lemma IV.6, we have the first term upper bounded by either

(i) $\underline{j=1}$: $\left(\frac{2^{\sigma \overline{S}'_1}}{\alpha}\right) \cdot \left(\frac{1}{q}\right)^{\left\lceil \frac{\lceil \gamma k \rceil}{B_1} \right\rceil}$ or,

(ii) $\underline{j=0}$: $\left(\frac{2^{\sigma \overline{S}'_0}}{(1-\alpha)}\right) \cdot \left(1-\frac{1}{q}\right)^{\left\lceil \frac{(\lfloor (1-\gamma)k \rfloor + 1)}{B_0} \right\rceil}$

To upper bound the second term, we invoke Markov inequality. If $w_{(x,y)}$ denotes the workspace required for computation on $(\boldsymbol{x},\boldsymbol{y})$, then $w_{(x,y)} = \log_2 [\max_{v \in P_{(x,y)}} l(v)]$ where $P_{(x,y)}$ is the unique computation path corresponding to $(\boldsymbol{x},\boldsymbol{y})$. So

$$\Pr(l(v) \geqslant 2^{\sigma \overline{S}'_j}, v \in P_{(x,y)}) = \Pr(w_{(x,y)} \geqslant \sigma \overline{S}'_j) \leqslant \frac{\overline{S}'_j}{\sigma \overline{S}'_j} = \frac{1}{\sigma}$$

We also know that $\overline{S}'_j \leqslant \overline{S}_j + \log_2 \lceil (1+\tau)\overline{T}_j \rceil$. Combining all of the above we get the following two inequalities,

(i) $\underline{j=1}$:

$$\log_2 \left[\alpha \cdot \left(\frac{\tau}{1+\tau} - \frac{1}{\sigma}\right)\right] + \left\lceil \frac{\lceil \gamma k \rceil}{\left\lceil \frac{\lceil (1+\tau)\overline{T}_1 \rceil}{d} \right\rceil} \right\rceil \cdot \log_2 q \leqslant \sigma(\overline{S}_1 + \log_2 \lceil (1+\tau)\overline{T}_1 \rceil) \tag{14}$$

(ii) $\underline{j=0}$:

$$\log_2 \left[(1-\alpha) \cdot \left(\frac{\tau}{1+\tau} - \frac{1}{\sigma}\right)\right] + \left\lceil \frac{\lfloor (1-\gamma)k \rfloor + 1}{\left\lceil \frac{\lceil (1+\tau)\overline{T}_0 \rceil}{d} \right\rceil} \right\rceil \cdot \log_2 \left(\frac{q}{q-1}\right)$$
$$\leqslant \sigma(\overline{S}_0 + \log_2 \lceil (1+\tau)\overline{T}_0 \rceil) \tag{15}$$

From Lemma IV.1 we get,

$$\overline{T}_1 \leqslant \frac{\overline{T}}{\alpha} \quad \text{and} \quad \overline{S}_1 \leqslant \frac{\overline{S}}{\alpha}$$
$$\overline{T}_0 \leqslant \frac{\overline{T}}{1-\alpha} \quad \text{and} \quad \overline{S}_0 \leqslant \frac{\overline{S}}{1-\alpha}$$

Now we use the above bounds in (14) and (15) to obtain:

$$\log_2\left[\alpha\cdot\left(\tfrac{\tau}{1+\tau}-\tfrac{1}{\sigma}\right)\right]+\left\lceil\frac{\lceil\gamma k\rceil}{\left\lceil\frac{\lceil(1+\tau)\overline{T}/\alpha\rceil}{d}\right\rceil}\right\rceil\cdot\log_2 q \leqslant \sigma\cdot\left(\tfrac{\overline{S}}{\alpha}+\log_2\left\lceil\tfrac{(1+\tau)\overline{T}}{\alpha}\right\rceil\right) \tag{16}$$

and,

$$\log_2\left[(1-\alpha)\cdot\left(\tfrac{\tau}{1+\tau}-\tfrac{1}{\sigma}\right)\right]+\left\lceil\frac{\lfloor(1-\gamma)k\rfloor+1}{\left\lceil\frac{\lceil(1+\tau)\overline{T}/(1-\alpha)\rceil}{d}\right\rceil}\right\rceil\cdot\log_2\left(\tfrac{q}{q-1}\right)$$
$$\leqslant \sigma\cdot\left(\tfrac{\overline{S}}{(1-\alpha)}+\log_2\left\lceil\tfrac{(1+\tau)\overline{T}}{(1-\alpha)}\right\rceil\right) \tag{17}$$

Each row of G is of weight at least d, thus any question of the form (12) or (13) can be answered only if $t_{(x,y)} \geqslant d+1$. This means $\overline{T} \geqslant d+1$. After some algebra, the tradeoffs implied by (16) and (17) for the branching program $\mathcal{B}$ are found to be as in $(*)$ in Table III on page 14.

Table III *Intermediate steps during the derivation of (11) from (16) and (17) in the proof of Theorem IV.1.*

$$\begin{array}{rcl}\left(\frac{\sigma(2+\tau)}{\alpha^2\gamma}\right)\cdot\overline{T}\cdot\left(\overline{S}+\alpha\cdot\log_2\left[\frac{(2+\tau)\overline{T}}{\alpha}\right]-(\frac{\alpha}{\sigma})\cdot\log_2\left[\alpha\cdot\left(\frac{\tau}{1+\tau}-\frac{1}{\sigma}\right)\right]\right) & \geqslant & kd\log_2 q \\ \left(\frac{\sigma(2+\tau)}{(1-\alpha)^2(1-\gamma)}\right)\cdot\overline{T}\cdot\left(\overline{S}+(1-\alpha)\cdot\log_2\left[\frac{(2+\tau)\overline{T}}{1-\alpha}\right]-(\frac{1-\alpha}{\sigma})\cdot\log_2\left[(1-\alpha)\cdot\left(\frac{\tau}{1+\tau}-\frac{1}{\sigma}\right)\right]\right) & \geqslant & kd\log_2\left(\frac{q}{q-1}\right)\end{array} \tag{†}$$

If we let $\zeta \stackrel{def}{=} \frac{(2+\tau)}{\left(\frac{\tau}{1+\tau}-\frac{1}{\sigma}\right)^{1/\sigma}}$ then (†) reduces to:

$$\begin{array}{rcl}\sigma(2+\tau)\overline{T}\left(\overline{S}+\alpha\cdot\log_2\overline{T}-(1+\frac{1}{\sigma})\alpha\log_2\alpha+\alpha\log_2\zeta\right) & \geqslant & kd\alpha^2\gamma\log_2 q \\ \sigma(2+\tau)\overline{T}\left(\overline{S}+(1-\alpha)\cdot\log_2\overline{T}-(1+\frac{1}{\sigma})(1-\alpha)\log_2(1-\alpha)+(1-\alpha)\log_2\zeta\right) & \geqslant & kd(1-\alpha)^2(1-\gamma)\log_2\left(\frac{q}{q-1}\right)\end{array} \tag{$*$}$$

The bound of $(*)$ may be optimized with respect to the proof parameters τ and σ for a specific γ subject to $\tau > 0$ and $\sigma > 1+\frac{1}{\tau}$. For example we chose $\tau = \sqrt{2}$ and $\sigma = \frac{7}{4}$. Finally observing that when $0 < \beta < 1$, $-\beta\log_2(\beta) \leqslant \frac{1}{\ln 2^e}$ we get the claimed tradeoff of (11). ■

Corollary IV.1 *Let* C *be an* (n,k,d) *linear code from a family* $\mathcal{F}$ *of asymptotically good codes over* F_q*. Let the decision function* $f_{\mathrm{C},\gamma}$ *be computable in the* q*-way branching program model in expected-time* $\overline{T}$ *using expected-space* $\overline{S}$*. Then the sequence of such branching programs must satisfy the asymptotic time-space tradeoff:*

$$\overline{TS} = \Omega\left(\tfrac{n^2}{q}\right) \tag{18}$$

Proof. Either α or $(1-\alpha)$ is at least $\frac{1}{2}$. Also γ is an absolute constant between 0 and 1. The family $\mathcal{F}$ is asymptotically good which means $k = \Theta(n)$ and $d = \Theta(n)$. Further, observe that $\log_2 q \geqslant \log_2\left(\frac{q}{q-1}\right)$. Expanding as a power-series, $\log_2\left(\frac{q}{q-1}\right) = -\log_2\left(1-\frac{1}{q}\right) = \sum_{m=1}^{\infty}\frac{1}{mq^m\ln 2}$. Thus $\frac{2}{q} \geqslant \log_2\left(\frac{q}{q-1}\right) > \frac{1}{q}$ since $q \geqslant 2$. So we get (18) and therefore Theorem IV.1 implies a quadratic tradeoff when $q = \Theta(1)$. ■

Corollary IV.1 implies quadratic time-space tradeoffs for Boolean decision branching programs verifying certain fundamental algebraic functions. Using algebraic code constructions, it is also possible to demonstrate a constructive decision function which has a quadratic time-space tradeoff in the Boolean decision branching program model.

V. A Constructive Decision Function with Quadratic Time-Space Product

In this section we present an explicit constructive decision function which has a quadratic time-space product in the Boolean decision branching program model. The function is related to the class of constructive binary codes called *Justesen codes* [Jus72]. Justesen codes are instances of concatenated codes which are asymptotically good. The following definition is based on the exposition in [MS77].

Definition V.1 *Let $\mathscr{R}$ be an $[N = 2^m - 1, K, D = N - K + 1]$ Reed-Solomon code over F_{2^m} and let α be a primitive element of F_{2^m}. Let $\boldsymbol{a} = (a_0, a_1, \ldots, a_{N-1})$ be a codeword from $\mathscr{R}$, such that $a_i \in \mathrm{F}_2^m$. Let $\boldsymbol{b}$ be the vector $(a_0, a_0;\ a_1, \alpha a_1;\ a_2, \alpha^2 a_2;\ \ldots;\ a_{N-1}, \alpha^{N-1} a_{N-1})$. Replace each of the $2N$ components of $\boldsymbol{b}$ by the corresponding binary m tuple to get a binary vector $\boldsymbol{c}$ of length $2mN$. For any N and K, with $0 < K < N$, the **Justesen code** $\mathscr{J}_{N,K}$ consists of all such vectors $\boldsymbol{c}$ which are obtained from $\mathscr{R}$.*

It is clear from the definition that $\mathscr{J}_{N,K}$ is a rate $R \stackrel{def}{=} \frac{K}{2N} \leqslant \frac{1}{2}$ linear binary code of length $n \stackrel{def}{=} 2mN$. It is also well known that the minimum distance of $\mathscr{J}_{N,K}$ is $d \geqslant n(1-2R)(H_2^{-1}(\frac{1}{2}) - o(1)) \approx 0.11n(1 - \frac{K}{N})$. See [MS77] for a proof. Invoking Corollary IV.1, we have proved the following:

Theorem V.1 *Let $0 < \gamma < 1$ and $0 < R < \frac{1}{2}$ be absolute constants and let the decision function $f_{\mathscr{J}_{N,K},\gamma}$ be computable in the 2-way branching program model in expected-time $\overline{T}$ using expected-space $\overline{S}$. Then the sequence of such branching programs must satisfy the asymptotic time-space tradeoff $\overline{TS} = \Omega(n^2)$.*

VI. Quadratic Time-Space Tradeoffs for Verifying Fundamental Algebraic Operations

In Section IV we considered the problem of computing $f_{\mathrm{C},\gamma}$ to partially verify the dual syndrome-vector of a q-ary linear code $\mathrm{C}_q[n,k,d]$. We showed that any q-ary branching program computing $f_{\mathrm{C},\gamma}$ must satisfy the time-space tradeoff:

$$6\overline{T}\left(\overline{S} + \log_2 \overline{T} + 7\right) \geqslant kd \cdot \max\left\{\alpha^2\gamma \log_2 q,\ \ (1-\alpha)^2(1-\gamma)\log_2 \tfrac{q}{q-1}\right\} \tag{19}$$

where α is the acceptance ratio of $f_{\mathrm{C},\gamma}$ which is related to γ and k.

We also showed that for a family of asymptotically good codes, branching programs which compute $f_{\mathrm{C},\gamma}$ must satisfy the asymptotic time-space tradeoff:

$$\overline{TS} = \Omega\left(\tfrac{n^2}{q}\right) \tag{20}$$

Now we will use the above results to derive lower bounds on the time-space resources required for verifying several fundamental operations in the branching program model. Target functions considered are: cyclic convolution, matrix-vector multiplication, and the discrete Fourier transform. These are defined precisely below:

Definition VI.1 *Let $\langle a(X)\rangle_2$ denote the binary vector of length n representing the detached coefficients of a polynomial $a(X)$ in the ring $\mathrm{F}_2[X]/(X^n - 1)$. Given a fixed polynomial $b(X)$ in this ring, the **n-bit convolution function** $CONV_{b(X)} : \{0,1\}^n \to \{0,1\}^n$ is defined by $CONV_{b(X)}\left(\langle a(X)\rangle_2\right) = \langle a(X) \odot b(X)\rangle_2$, where $\odot$ denotes polynomial multiplication over F_2 modulo $X^n - 1$.*

Definition VI.2 *Let $M_{n,n}$ denote the set of all $n \times n$ binary matrices. Given a fixed binary matrix $B \in M_{n,n}$, the **n-bit matrix-vector product function** $MVMUL_B : \{0,1\}^n \to \{0,1\}^n$ is defined by $MVMUL_B(\boldsymbol{x}) = \boldsymbol{x}B$, where $\boldsymbol{x}B$ denotes the usual (row) vector-matrix product operation over F_2.*

Definition VI.3 *Let $n = 2^m$. For a vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-2})$ over F_n, let $\langle \boldsymbol{x} \rangle_2$ denote the binary vector of length mn obtained by concatenating the binary representations of $x_0, x_1, \ldots, x_{n-2}$ with respect to a fixed basis for F_n over F_2. Given a primitive element α of F_n and a fixed fraction $\rho = k/(n-1)$ in $(0,1)$, the* ***k-point DFT function*** *$DFT_{\rho,\alpha} : \{0,1\}^{m(n-1)} \to \{0,1\}^{\rho m(n-1)}$ is*

$$DFT_{\rho,\alpha}\Big(\langle \boldsymbol{x} \rangle_2\Big) \stackrel{def}{=} \Big\langle \big(\sum_{i=0}^{n-2} x_i, \sum_{i=0}^{n-2} x_i \alpha^i, \ldots, \sum_{i=0}^{n-2} x_i \alpha^{ji}, \ldots, \sum_{i=0}^{n-2} x_i \alpha^{(k-1)i}\big) \Big\rangle_2$$

The following definition fixes the notion of *partial verification* of a vector valued function:

Definition VI.4 *Let $\mathcal{D}$ be a finite domain. Consider a vector valued function $f : \mathcal{D}^n \to \mathcal{D}^k$. Let γ be an absolute real constant such that $0 < \gamma < 1$. For an input $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}^n \times \mathcal{D}^k$ the decision function $\zeta_{f,\gamma} : \mathcal{D}^n \times \mathcal{D}^k \to \{0,1\}$ evaluates true, namely $\zeta_{f,\gamma}(\boldsymbol{x}, \boldsymbol{y}) = 1$ if and only if at least a fraction γ of the k equations $f_i(\boldsymbol{x}) = y_i$ are satisfied. Here $f_i(\cdot)$ denotes the i^{th} coordinate of the evaluated function.*

Our results for unrestricted deterministic decision branching programs partially verifying fundamental operations are summarized in Table II. The detailed proofs are given below. We have the following results:

Theorem VI.1 *Let γ be an absolute real constant such that $0 < \gamma < 1$. A deterministic boolean decision branching program $\mathcal{B}$ computing either $\zeta_{\mathrm{CONV},\gamma}$ or $\zeta_{\mathrm{MVMUL},\gamma}$ in expected time $\overline{T}$ and expected space $\overline{S}$ satisfies: $\overline{TS} = \Omega(n^2)$*

Proof. Let $\mathrm{C}(2n, n, d)$ be a rate half systematic quasi-cyclic code with generator matrix of the form $G = [I_n \mid C]$, where C is a square $n \times n$ circulant matrix. A result due to Chen, Peterson and Weldon [CPW69] and Kasami [Kas74] says that the family of rate half quasi-cyclic codes achieve the binary GV bound. Moreover they show that this occurs when the polynomial associated with the first row of C is relatively prime to $(x^n - 1)$. It is also clear that the columns of the parity check matrix can be rearranged to get G. Therefore the dual code $\mathrm{C}^{\perp}$ is equivalent to C.

Let the binary polynomial associated with the circulant matrix C be $g(x)$. The encoding of a binary information polynomial $i(x) \in GF(2)[x]/(x^n - 1)$ can then be represented in the following form:

$$i(x) \mapsto c(x) = (i(x)\ ,\ i(x) \odot g(x))$$

where $\odot$ represents CONV operation.

So as to arrive at a contradiction, let us suppose that $\zeta_{\mathrm{CONV},\gamma}$ has a time-space tradeoff given by $\overline{TS} = o(n^2)$ in the branching program model. A trivial modification of such a branching program will compute $f_{\mathrm{C}}^{\perp}$ in $\overline{TS} = o(n^2)$. To see how, consider an input $(\boldsymbol{x}, \boldsymbol{y}) \in \mathrm{F}_q^{2n} \times \mathrm{F}_q^n$ and observe that $\boldsymbol{g}_i \cdot \boldsymbol{x} = x_i + \tilde{\boldsymbol{c}}_i \cdot \tilde{\boldsymbol{x}}$, where $\boldsymbol{g}_i$ is the i^{th} row of the generator matrix G and $\boldsymbol{c}_i$ is the i^{th} row of the circulant matrix C. Here $\tilde{\boldsymbol{x}}$ denotes the vector in F_q^n formed by taking only the last n coordinates of $\boldsymbol{x}$. The term $\tilde{\boldsymbol{c}}_i \cdot \tilde{\boldsymbol{x}}$ forms a part of the CONV operation. This means $f_{\mathrm{C}}^{\perp}$ can be implemented using $\zeta_{\mathrm{CONV},\gamma}$ which operates on $(\boldsymbol{x}, \boldsymbol{y} - \hat{\boldsymbol{x}})$, where $\hat{\boldsymbol{x}}$ denotes the vector in F_q^n formed by taking only the first n coordinates of $\boldsymbol{x}$. Finally, since by (20), $f_{\mathrm{C}}^{\perp}$ computation satisfies $\overline{TS} = \Omega(n^2)$ in the branching program model, so should $\zeta_{\mathrm{CONV},\gamma}$.

To obtain the time-space lower bound for $\zeta_{\mathrm{MVMUL},\gamma}$, consider the problem of computing $G\boldsymbol{x}$, given an input vector $\boldsymbol{x}$, where G represents the generator matrix of a random linear code. As the family of linear codes is asymptotically good, by (20), we get a time-space tradeoff for $\zeta_{\mathrm{MVMUL},\gamma}$. ■

Theorem VI.2 *Let γ be an absolute real constant such that $0 < \gamma < 1$. Let $n = 2^m$ be an integer. Let $0 < \rho < 1$ be a fixed fraction and let $k = \rho(n-1)$. A deterministic boolean decision branching program*

$\mathcal{B}$ which partially verifies a k-point DFT by computing $\zeta_{\mathrm{DFT},\gamma}$ in expected time $\overline{T}$ and expected space $\overline{S}$ satisfies: $\overline{T}\overline{S} = \Omega(n^2 \log_2 n)$.

Proof. As in the proof of Theorem VI.2, consider a length $(n-1)$, dimension k Reed-Solomon code, $\mathcal{R}$. Let the roots of its generating polynomial in F_{2^m} be the $(n-1)$ consecutive powers α^j of a primitive element $\alpha \in \mathrm{F}_{2^m}$. Both $\mathcal{R}$ and $\mathcal{R}^\perp$ are MDS codes and have fixed rates. The syndrome computation for $\mathcal{R}^\perp$ is precisely the computation of a k-point DFT.

Now take the binary image code of $\mathcal{R}$, which we denote by C. C is a $((n-1)m, km, n-k)$ linear binary code and $\zeta_{\mathrm{DFT},\gamma}$ is precisely $f_{\mathrm{C},\gamma}$. The result follows on applying (19). ■

VII. Summary

We derived a minimum distance bound for codes using the deterministic branching program model for non-uniform sequential computation of a decision function related to code-coset membership. Specifically, we looked at the deterministic branching program complexity of verifying the syndrome vector of linear codes. We derived the first ever quadratic time-space bounds for unrestricted deterministic decision branching programs.

The minimum distance bounds so derived were used to obtain the first ever time-space tradeoffs for unrestricted deterministic decision branching programs verifying several fundamental operations. These results are derived making use of deep new connections between the properties of certain algebraic codes and fundamental algorithms.

VIII. Acknowledgments

Funding for this work was provided in part by the National Science Foundation and the California Institute of Telecommunications and Information Technology at the University of California, San Diego. The material in this paper is being prepared for submission in a future conference and journal.

References

[Abr91] K. Abrahamson. "Time-space tradeoffs for algebraic problems on general sequential machines," *Journal of Computer and System Sciences*, **43**:269–289, 1991.

[Ajt98] M. Ajtai. "A nonlinear time lower bound for boolean branching programs," In *Proceedings of the 40-th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 60–70, New York, NY, October 1998.

[Ajt99] M. Ajtai. "Determinism versus non-determinism for linear time RAMs with memory restrictions," In *Proceedings of the 31-st Annual ACM Symposium Theory of Computing (STOC)*, pp. 632–641, Atlanta, GA, May 1999.

[BC82] A. Borodin and S. A. Cook. "A time-space tradeoff for sorting on a general sequential model of computation," *SIAM Journal on Computing*, **11**:287–297, 1982.

[BJS01] P. Beame, T. S. Jayram, and M. Saks. "Time-space tradeoffs for branching programs," *Journal of Computer and System Sciences*, **63**:542–572, 2001.

[BM05] L. M. J. Bazzi and S. K. Mitter. "Encoding complexity versus minimum distance," *IEEE Transactions on Information Theory*, **IT-51**:2103–2112, June 2005.

[BRS83] A. Borodin, A. A. Razborov, and R. Smolensky. "On lower bounds for read-k-times branching programs," *Computational Complexity*, **3**:1–18, 1983.

[BSS03] P. Beame, M. Saks, X. Sun, and E. Vee. "Time-space tradeoff lower bounds for randomized computation of decision problems," *Journal of the ACM*, **50**:154–195, 2003.

[BST98] P. Beame, M. Saks, and J. S. Thathachar. "Time-space tradeoffs for branching programs," In *Proceedings of the 39-th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 254–263, Palo Alto, CA, November 1998.

[BV02] P. Beame and E. Vee. "Time-space tradeoffs, multiparty communication complexity, and nearest-neighbor problems," In *Proceedings of the 34-st Annual ACM Symposium Theory of Computing (STOC)*, pp. 688–697, Montreal, Quebec, Canada, May 2002.

[BW01] B. Bollig and P. Woelfel. "A read-once branching program lower bound of $\Omega(2^{n/4})$ for integer multiplication using universal hashing," In *Proceedings of the 33-st Annual ACM Symposium Theory of Computing (STOC)*, pp. 419–424, Heronissos, Crete, Greece, July 2001.

[CPW69] C. L. Chen, W. W. Peterson, and E. J. Weldon Jr. "Some results on quasi-cyclic codes," *Information and Control*, **15**:407–423, 1969.

[Ger94] J. GERGOV. "Time-space tradeoffs for integer multiplication on various types of input oblivious sequential machines," *Information Processing Letters*, **51**:265–269, 1994.

[Juk95] S. JUKNA. "The graph of integer multiplication is hard for read-k-times networks," Technical Report 95-10, Universität Trier, 1995.

[Juk02] S. JUKNA. "On nondeterministic linear time branching programs," available at http://lovelace.thi.informatik.uni-frankfurt.de/~jukna/ftp/paul1.pdf, 2002.

[Jus72] J. JUSTESEN. "A class of constructive asymptotically good algebraic codes," *IEEE Transactions on Information Theory*, **IT-18**:652–656, July 1972.

[Kas74] T. KASAMI. "A Gilbert-Varshamov bound for quasi-cyclic codes of rate $1/2$," *IEEE Transactions on Information Theory*, **IT-20**:679–681, 1974.

[Kuz76] V. A. KUZ'MIN. "An estimate of the complexity of the realization of functions of the algebra of logic by the simplest forms of binary programs," *Metody Diskretnogo Analiza*, **29**:11–39, July 1976.

[Lee59] Y. LEE. "Representation of switching circuits by binary-decision programs," *Bell Systems Technical Journal*, **IT-38**:985–999, 1959.

[MNT93] Y. MANSOUR, N. NISAN, AND P. TIWARI. "The computational complexity of universal hashing," *Theoretical Computer Science*, **107**:121–133, 1993.

[Mor73] J. MORGENSTERN. "Note on a lower bound of the linear complexity of the fast Fourier transform," *Journal of the ACM*, **20**:305–306, 1973.

[MS77] F. J. MACWILLIAMS AND N. J. A. SLOANE. *The Theory of Error Correcting Codes*. North-Holland/Elsevier, Amsterdam, 1977.

[Oko91] E. A. OKOL'NISHNIKOVA. "Lower bounds for branching programs computing characteristic functions of binary codes," *Metody Diskretnogo Analiza*, **51**:61–83, 1991. (in Russian).

[Oko02] E. A. OKOL'NISHNIKOVA. "On one lower bound for branching programs," *The Electronic Colloquium on Computational Complexity (ECCC)*, **20**, 2002. Available at http://www.eccc.uni-trier.de/eccc-reports/2002/TR02-020.

[Pag01] J. PAGTER. *Time-Space Tradeoffs*. PhD thesis, University of Aarhus, Denmark, March 2001.

[Pip78] N. PIPPENGER. "A Time-space Tradeoff," *Journal of the ACM*, **25**:509–515, 1978.

[Pon98] S. PONZIO. "A lower bound for integer multiplication with read-once branching programs," *SIAM Journal on Computing*, **26**:798–815, 1998.

[Raz91] A. A. RAZBOROV. "Lower bounds for deterministic and non-deterministic branching programs," *Lecture Notes in Computer Science (LNCS)*, **529**:47–60, July 1991.

[San06] N. SANTHI. *Graphical Models for Coding and Computation*. PhD thesis, University of California, San Diego, USA, 2006. In preparation.

[SV06] N. SANTHI AND A. VARDY. "Minimum Distance of Codes and their Branching Program Complexity," In *Proceedings of the International Symposium on Information Theory (ISIT)*, pp. 1490–1494, July 2006.

[SW03] M. SAUERHOFF AND P. WOELFEL. "Time-space tradeoff lower bounds for integer multiplication and graphs of arithmetic functions," In *Proceedings of the 35-st Annual ACM Symposium Theory of Computing (STOC)*, pp. 186–195, San Diego, CA, June 2003.

[Weg87] I. WEGENER. *The Complexity of Boolean Functions*. Wiley, New York, NY, 1987.

[Weg00] I. WEGENER. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. SIAM Press, Philadelphia, PA, 2000.

[Win67] S. WINOGRAD. "On the time required to perform multiplication," *Journal of the ACM*, **14**:793–802, 1967.

[Yes84] Y. YESHA. "Time-space tradeoffs for matrix multiplication and the discrete Fourier transform," *Journal of Computer and System Sciences*, **29**:183–197, 1984.